TRIANGLE RENDERING USING DIRECT EVALUATION

[0001] This application claims priority from U.S. Provisional Application Serial No. 60/455,923, filed March 18, 2003, the entire content of which is incorporated herein by reference.

TECHNICAL FIELD

[0002] The invention relates to mobile computing devices and, in particular, to techniques for rendering graphics for display by the mobile computing devices.

BACKGROUND

[0003] Mobile computing devices, such as personal digital assistants (PDAs), wireless communication devices, global positioning devices, and the like, are increasingly requiring advanced two-dimensional (2D) and three-dimensional (3D) graphics applications. For example, the mobile computing devices increasingly offer games, character animations, graphical menu systems, and other applications that require advanced graphics rendering. This advanced graphics rendering, especially with 3D graphics, involves a substantial amount of data processing. Furthermore, these processing requirements only increase as the mobile computing devices adopt display panels that offer increased resolution and color depth.

[0004] One fundamental operation of a 3D graphics application is triangular rendering. In particular, a 3D graphical object is typically defined by a set of triangles in a 3D space. Each triangle is defined by three vertices within the 3D space and associated attributes of each vertex, including color attributes, texture coordinates, or both. Each triangle is graphically rendered using an interpolation process that fills the interior of the triangle based on the location and attribute information defined at the three vertices. More specifically, each starting and ending coordinate is computed for each line of pixels within the triangle using the interpolation process. The pixels along each line are sequentially filled to render the entire triangle.

[0005] One challenge for mobile computing devices, however, is the typical limitation of available power. Unlike a desktop computing environment, mobile computing devices

typically have limited battery power. As a result, the computationally intensive interpolation process required for triangular rendering may place a significant strain on the available power of the mobile computing device.

SUMMARY

[0006] The disclosure is directed to techniques for rendering a triangular area of pixels for presentation on a display device. The techniques may be particularly useful in a mobile computing environment presenting limited power resources.

[0007] In one embodiment, a method comprises computing data that defines a rectangular area of pixels that bounds a triangular area of the pixels, and evaluating coordinates of the pixels of the rectangular area to determine which pixels fall within the triangle area. The method further comprises updating pixel data for the pixels that fall within the triangle area to render the triangular area.

[0008] In another embodiment, an apparatus comprises a rendering engine that defines a rectangular area of pixels that bounds a triangular area of the pixels. The rendering engine evaluates coordinates associated with the pixels of the rectangular area to selectively render the pixels that fall within the triangular area.

[0009] In another embodiment, a mobile communication device comprises a display, a processor, and a rendering engine. The processor generates video output data for presentation by the display as a graphical environment. The rendering engine applies a direct evaluation algorithm to render a triangle for the graphical environment, wherein the direct evaluation algorithm applies linear equations to render the triangle without interpolating between edges of the triangle.

[0010] The details of one or more embodiments of the invention are set forth in the accompanying drawings and the description below. Other features, objects, and advantages of the invention will be apparent from the description and drawings, and from the claims.

BRIEF DESCRIPTION OF DRAWINGS

[0011] FIG. 1 is a block diagram illustrating an example embodiment of a mobile computing device.

[0012] FIG. 2 is a graph that illustrates application of direct evaluation triangle rendering techniques.

[0013] FIG. 3 is a flowchart illustrating example operation of a rendering engine in application of the direct evaluation techniques.

[0014] FIG. 4 is a block diagram illustrating an example embodiment of the rendering engine of FIG. 3.

DETAILED DESCRIPTION

[0015] FIG. 1 is a block diagram illustrating an example mobile computing device 2 that presents a three-dimensional (3D) graphical environment on display 4. Mobile computing device 2 may be, for example, a personal digital assistant (PDA), a wireless communication device such as a mobile phone or satellite phone, a global positioning device, a portable digital television, a digital video camera, a wireless video device, a device integrating any of the foregoing devices, and the like. Examples of the 3D graphical environment presented by mobile computing device 2 include games, character animations, graphical menu systems, and other applications that require advanced graphics rendering.

[0001] Processor 6 provides primary control over the components of mobile computing device 2. For example, processor 6 captures input from a user via input device 8, such as a keypad, and directs digital signal processor (DSP) 10 to generate video output data for presenting the 3D graphical environment on display 4. Display 4 may comprise any output device, e.g., a liquid crystal display (LCD) for a camcorder or a mobile phone screen. In particular, DSP 10 issues commands to rendering engine 12 to direct the rendering engine to render graphical objects, e.g., triangles, within a 3D space. Based on the commands, rendering engine 12 processes video data stored within video memory 14 and cache 15 for driving display 4.

[0002] Processor 6 may take the form of an embedded microprocessor, specialized hardware, software, e.g., a control software module, or combinations thereof. Moreover, DSP 10, processor 6, rendering engine 12, as well as other components of mobile computing device 2, may be implemented in one or more application-specific integrated circuits (ASICs), as multiple discrete components, or combinations thereof.

[0016] Memories 16 and 18 store instructions and data for use by processor 6 and DSP 10, respectively. Although illustrated separately, memories 16 and 18 may comprise one or more memory devices. Moreover, memories 16 and 18 may comprise read-only memory (ROM), synchronous dynamic random access memory (SDRAM), non-volatile static random access memory (SRAM), Flash memory, electrically erasable programmable read-only memory (EEPROM), and the like.

[0017] As described herein, DSP 10 and a rendering engine 12 interact to apply a direct evaluation technique to render triangles for the 3D graphical environment. More specifically, DSP 10 issues commands to rendering engine 12 to direct the rendering engine to render triangles within a 3D space. In response, rendering engine 12 applies a direct evaluation triangle rendering algorithm that requires fewer division operations than the more computationally intensive interpolation process employed by conventional systems. As a result, mobile computing device 2 may present a 3D graphical environment while preserving as much as possible the power available to the mobile computing device.

[0018] To invoke rendering engine 12, DSP 10 issues a command that specifies the three vertices of the triangle to be rendered. For example, as illustrated in the example triangle depicted by FIG. 2, DSP 10 issues a command that defines triangle 20 with a first vertex (X_0,Y_0), a second vertex (X_1,Y_1), and a third vertex (X_2,Y_2), where each of the vertices represents a coordinate within a 2D slice of a 3D environment. Consequently, the vertices define a triangle having edges 24A, 24B and 24C.

[0019] In response to the command, rendering engine 12 computes a "bounding box" 22, which refers to a rectangular area defined to bound a minimum and a maximum of the x,y coordinates of the triangle. Bounding box 22, for example, is defined by an upper-left coordinate (X_{min},Y_{min}) and a lower-right coordinate (X_{max},Y_{max}). As illustrated in the example of FIG. 2, bounding box 22 may be defined to cover a minimum area within the 2D slice to encompass triangle 20. In other embodiments, minimum dimensions for bounding box 22 may be computed as function of a block size of video blocks stored by cache 15. In particular, DSP 10 and rendering engine 12 may process output video data of video memory 14 in a "tiled" fashion, i.e., in block fashion along rows and columns. Moreover, the block size may be set based on the capacity of cache 15. For example, the block size may equal the capacity of the cache. In this manner, the dimensions of bounding box 22 may be defined to

be equal to or less than the dimensions of a video block in the event a triangle to be rendered lies completely within a single video block. For triangles spanning more than one video block, a bounding box may be used that has dimensions calculated to encompass the overlap between the bounding box of the triangle and the video block, as illustrated by bounding box 22A. This may advantageously reduce the number of memory accesses required for retrieving and rendering pixel data associated with the triangle, which translates into power savings.

[0020] After calculating the bounding box, e.g., bounding box 22 of FIG. 2, rendering engine 12 processes each pixel line encompassed by the bounding box and selectively enables those pixels that fall within triangle 20. Specifically, rendering engine 12 scans the pixels lines, e.g., in a leftward and downward fashion, and applies linear equations to determine whether each pixel falls within triangle 20. For those pixels within the edges 24A, 24B and 24C, rendering engine 12 updates the current attributes for the pixels, e. g., based on z-value, color information, texture information, or the like, associated with triangle 20.

[0021] FIG. 3 is a flowchart illustrating example operation of rendering engine 12 in application of a direct evaluation algorithm for rendering a triangle. Initially, rendering engine 12 receives a command from DSP 10 that specifies three vertices of a triangle to be rendered, i.e., a first vertex (X_0,Y_0), a second vertex (X_1,Y_1), and a third vertex (X_2,Y_2) (step 30 of FIG. 3).

[0022] In response to the command, rendering engine 12 computes data, e.g., an upper left coordinate and a height and width, that defines the bounding box that encompasses the triangle defined by the vertices (step 32).

[0023] In addition, rendering engine 12 computes a coefficient matrix M^{-1} for computing the linear coefficients of the linear equations that describe the attributes of each pixel within the triangle, and a coefficient matrix M_C for computing the linear coefficients of the linear equations that describe the three edges of the triangle (step 34). Specifically, an attribute value v for a pixel located inside the triangle can be represented as a linear function of the x,y location in the screen space as follows:

$$v = Ax + By + C. \qquad (1)$$

[0024] Consequently, the linear coefficients, A, B, and C for eq. (1) can be obtained from the three equations associated with the three vertices:

$$\begin{aligned} v_o &= Ax_o + By_o + C \\ v_1 &= Ax_1 + By_2 + C \\ v_2 &= Ax_2 + By_2 + C \end{aligned} \tag{2}$$

where $v_i (i = 0, 1, 2)$, is the attribute value defined at vertex i, and $(x_i, y_i)(i = 0, 1, 2)$ are the screen coordinates of vertex i. Using matrix notations, the vertices can be represented as:

$$\begin{bmatrix} v_o \\ v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} x_0 & y_0 & 1 \\ x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \end{bmatrix} = \begin{bmatrix} A \\ B \\ C \end{bmatrix}. \tag{3}$$

[0025] Thus, the coefficients for attribute interpolation can be expressed as:

$$\begin{bmatrix} A \\ B \\ C \end{bmatrix} = M^{-1} \begin{bmatrix} v_0 \\ v_1 \\ v_2 \end{bmatrix}. \tag{4}$$

where

$$M = \begin{bmatrix} x_0 & y_0 & 1 \\ x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \end{bmatrix} \tag{5}$$

[0026] The coefficient matrix M^{-1} for computing the coefficients for attribute interpolation can be computed using Laplace expansion as follows

$$M^{-1} = \frac{1}{\det(M)} M_C^T \tag{6}$$

where

$$\det(M) = |M| = x_1 y_2 + x_2 y_0 + x_0 y_1 - x_2 y_1 - x_0 y_2 - x_1 y_0 \tag{7}$$

M_c, the companion matrix of M, can be calculated as:

$$M_c = \begin{bmatrix} y_1 - y_2 & x_2 - x_1 & x_1 y_2 - x_2 y_1 \\ y_2 - y_0 & x_0 - x_2 & x_2 y_0 - x_0 y_2 \\ y_0 - y_1 & x_1 - x_0 & x_0 y_1 - x_1 y_0 \end{bmatrix}. \quad (8)$$

[0027] Upon calculating the coefficient matrices M^{-1} and M_C, rendering engine 12 computes coefficients A, B, C, for all attributes associated with the pixels (step 36). In particular, the coefficient matrix M^{-1} can be used for computing the coefficients of different attributes, although the values at the vertices, (v_0, v_1, v_2), are different for different attributes.

[0028] Next, rendering engine 12 initializes all attribute values at a starting pixel of the bounding box, e.g., (X_{min}, Y_{min}) of FIG. 2, (step 38). In addition, rendering engine 12 initializes current pixel index values (X_C, Y_C) to a first pixel of a first scan line of the bounding box, i.e., X_{MIN} and Y_{MIN} (step 39).

[0029] During the direct evaluation rasterization process, rendering engine 12 tests the current pixel (X_C, Y_C) to determine whether the pixel falls within the triangle (step 39). Specifically, rendering engine 12 applies the previously calculated matrix M_C based on a set of edge equations delineating the triangle to determine whether the location of the current pixel falls within the triangle. These equations are uniquely defined by the coordinates of the three vertices. In particular, for a given starting vertex (x_i, y_i), and ending point (x_j, y_j), on a 2D plane, for example, the edge equation is,

$$\frac{y - y_i}{x - x_i} = \frac{y_j - y_i}{x_j - x_i} \quad (9)$$

or,

$$(y_i - y_j)x + (x_j - x_i)y + x_i y_j - x_j y = 0. \quad (10)$$

For the three edges with starting vertex index, $i = 1, 2, 0$, and ending vertex index, $j = 2, 0, 1$, three edge equations in matrix format can be expressed as:

$$\begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = M_C \begin{bmatrix} X_C \\ Y_C \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}. \qquad (11)$$

Thus, rendering engine 12 applies the matrix M_C to the current pixel (X_C, Y_C), and determines that the pixel is inside the triangle when all of e_1, e_2, e_3 are less than zero.

[0030] If the pixel is not within the triangle (no branch of 39), rendering engine 12 proceeds to the next pixel along the scan line of the bounding box, i.e., increments X_C (step 46) and, assuming the end of the scan line has not been reached (step 48), tests that pixel in accordance with equation (11) to determine whether the pixel is within the triangle (step 39).

[0031] If application of equation (11) indicates that the current pixel is within the triangle, rendering engine 12 determines whether the current pixel is visible (step 40). Specifically, rendering engine 12 determines the position of the current pixel along the z-axis to determine whether the pixels is visible, or whether pixels from other slices of the 3D graphic environment prevent the pixel from being visible to the user, i.e., are on top of the current pixel. In one embodiment, rendering engine 12 compares a z-value, z_c, of the current pixel with the corresponding z-value, z_b, of a z-buffer to determine if the pixel is visible, i.e., if $z_c < z_b$.

[0032] If the pixel is not visible (no branch of step 40), rendering engine 12 proceeds to the next pixel (step 46). If, however, the pixel is visible, rendering engine 12 computes attribute values, e.g., texture and color values, for the pixel based on equation (1) (step 42). Upon computing the attribute values, rendering engine 12 updates the value of the current pixel within video memory 14 based on the computed attributes (step 44).

[0033] After updating the value of the current pixel within video memory 14, rendering engine 12 proceeds to the next pixel (step 46). This process is repeated until the end of the current scan line is reached, i.e., $X_C > X_{MAX}$ or the current pixel exits the triangle (step 48), causing rendering engine 12 to proceed to the beginning of the next scan line, i.e., set $X_C=X_{MIN}$ and incrementing Y_C (step 50). Rendering engine 12 traverses each scan line of the bounding box testing each pixel in this manner, and terminates the rendering process when the last scan line has been completed, i.e., $Y_C>Y_{MAX}$ (step 52).

[0034] FIG. 4 is a block diagram illustrating an example embodiment of rendering engine 12. In the illustrated embodiment, rendering engine 12 includes a vertex buffer 60 for buffering vertices for triangles to be rendered in accordance with the direct evaluation techniques. In particular, rendering engine 12 receives commands from DSP 10 that specify triangles in the form of three vertices. Each vertex is defined by at least an x and y coordinate, and may include one or more attribute values k, such as z-values, color values, texture values, and the like.

[0035] Rendering engine 12 queues the vertex information in vertex buffer 60 for processing by processing engine 61, which includes bounding box generator 62, edge coefficient generator 64, and attribute coefficient generator 66. Bounding box generator 62 computes and outputs data values that define the dimensions of the bounding box associated with the triangle to be rendered. For example, bounding box generator 62 may output an upper-left coordinate (X_{min},Y_{min}) and a lower-right coordinate (X_{max},Y_{max}) for the bounding box.

[0036] Edge coefficient generator 64 processes the vertex coordinate information to generate the linear edge coefficients matrix M_c, as described above. Similarly, attribute coefficient generator 66 processes the vertex coordinate information to generate the linear attribute coefficients A, B, C as described above.

[0037] Rendering engine 12 further includes rasterizer 74 that processes the parameters produced by processing engine 61, and applies the direct evaluation techniques to produce pixel data to drive video memory 14 and display 4. Output buffer 63 includes bounding box buffer 68, edge coefficient buffer 70, and attribute coefficient buffer 72 to queue parameters defining the bounding box, linear edge coefficients, and linear attribute coefficients associated with each triangle to be rendered by rasterizer 74.

[0038] Various embodiments of the invention have been described. These and other embodiments are within the scope of the following claims.